Extracting Information from Point Set for Robust Fingerprint Authentication

Shen Ren

Bachelor of Computing in Computer Science

National University of Singapore

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2006

To my parents.

Abstract

In biometric identification, a fingerprint is typically represented as a set of minutiae which are 2-D points. We are interested in ways of extracting information from such a point set template for robust authentication. This thesis consists of two parts. In the first part, we show that analysis of an existing method, *Fingerprint vault*[6], is inadequate. As such, the security of such method may be too weak in practice. More specifically, we propose an attacker who achieves 2192.7 times speedup compared with a brute-force attacker whom they used to measure the security of fingerprint vault. In the second part, we propose a new method to extract information from 2-D point set by using feature lines and PCA transformation. We name this scheme as *approximate message authentication code for 2-D point set* to produce MAC like binary code for robust fingerprint authentication. Given such a 128-bit length AMAC code, to successfully launch a pre-image attack requires more than 2^{81} trials.

Acknowledgments

I would like to thank my research supervisor Dr. Chang Ee-Chien for his invaluable guidance, suggestions, and support throughout the course of this thesis.

I would like to thank my ex-supervisor Dr. Tay Yong Chiang for his important discussion.

I also want to take this opportunity to thank my fellow lab mates. They have offered their generous help and support to my research.

I am deeply grateful for my parents. Their love accompanies and encourages me every moment. I would like to dedicate this work to them.

Contents

Chapter 1

Introduction

Fingerprint templates, as well as many other biometric templates, are typically expressed as 2-D point sets, see Figure 1.1. Since every different scan of a same finger probably gives similar but not exactly the same point set, extracting consistent secret bits from the point set is difficult. On the other hand, consistent secret bits is a pre-requisite for cryptographic applications. For example, using the fingerprint as the symmetric secret key during encryption. Hence, it is an important problem to study the methods of extracting consistent secret bits from 2-D point sets.

1.1 Background

Authentication

According to the definition of Bishop[3], Authentication is the binding

of an identity to a subject. There are three classes of functions could be used to produce authentication information: *message encryption* , *message authentication code* and *hash function*[18]. A *message authentication code* is a public function of the message and a secret key which produces a fixed length value which serves as the identity information in authentication. More specifically, if using *message authentication code* (MAC) as the authentication function, we may have following authentication process: Given a document, say D, which is supposingly originated from Alice. MAC is an additional piece of information appended to D, and to facilitate authenticity verification. Typically, Alice would compute a MAC based on D and a key K_{Alice}

$$MAC = Encode(D, K_{Alice})$$

Now, given MAC and D, Bob wants to check whether D is indeed originated from Alice. He runs the verification using key K_{Bob}. Under the symmetric scenarios, K_{Bob} is the same as K_{Alice}. The verification output `Accept` or `Reject`.

$$Verification(D, MAC, K_{Bob})$$

There are a few requirements for this authentication process:

1. If the MAC is indeed computed by Alice, the verification must output `ACCEPT`.
2. MAC should be short to be easy for verification.
3. If $MAC \neq Encode(D, K_{Alice})$, the verification should output `reject`

with high probability.

Authentication using Fingerprints on Smart-card

Recently, a technique of combining fingerprints and smart-cards is applied to individual IC. The technique prints a clear version of fingerprint X on IC together with it's hash version $H(X)$ stored inside the card. Upon authentication process, the fingerprint X' printed on smart-card is scanned through a machine and feeded to a hash function. If the output hash value $H(X')$ matches the value $H(X)$ stored in the smart-card, the authentication process outputs `YES`.

If a malicious user obtains such an IC of someone S, he is able to change the fingerprint X printed on the card to his own Y, but he is not able to hack into the card to change the hash value $H(X)$. When he presents this card to an authority agency A to pretend to be S, A calculates $H(Y)$ and finds $H(Y) \neq H(X)$. Hence A could immediately reject the malicious user.

Noises in Fingerprint Templates

Authentication in fingerprint templates, as well as most of the biometric templates, is different from traditional cryptographic authentication because of the noises introduced in different places. For example, in fingerprint context, each scan of a same finger might be different possibly due to the hardness of your press, the cleanliness of your finger and many other reasons. Hence, from each scan you might get slightly different minutiae set.

There are mainly two kinds of noises in fingerprint, *white noises* and *replacement noises*, See Figure 1.1. [1]:

- *White noises* means during fingerprint scanning process, the minutiae points may perturb to a nearby location.
- *Replacement noise* means during the fingerprint scanning process, some original minutiae may not be captured and some new minutiae may be introduced.

(a) (b)

Figure 1.1: (a) The original fingerprint. The dots are the extracted minutiae. (b) The dots are the original minutiae. The "+" are minutiae extracted from another scan of the same finger.

A typical minutiae set will consists of around 30 minutiae points. A noise version of minutiae set may introduce around 10% of the replacement noise and 90% of the white noise. The white noise usually follows Gaussian distribution with very small variance.

[1]Fingerprint raw image is obtained from [1]

The traditional cryptographic authentication schemes don't take into account that the input documents may contain permissive amount of noises. In addition, we find in fingerprint template and many other 2-D point sets, the noises are closely related to the positioning information of each point, which is not captured by the classic cryptographic authentication schemes either.

Therefore in designing authentication function for noisy fingerprint data, we need to ensure the scheme has two more abilities than a classic cryptographic authentication function, i.e. *Robustness* and *Sensitivity*:

1. *Robustness* is the ability to tolerate permissive amount of noises from the input templates;

2. *Sensitivity* is the ability to detect the templates which have been tampered above some noisy threshold.

We further notice that authentication using biometric information has been studied for a long time, however, extracting authentication information from biometric information is relatively a new chapter.

Existing Methods to Handle Noisy Data

The most genetic approach [8] to do robust authentication is to use construction of *secure sketch/fuzzy extractor*. They could produce locality preserving hashes which tolerate certain amount of noises while still being sensitive to large factor of difference. BCH(Bose Chaudhuri Hocquenghem)[13] code is initialized for error correction in their proposal.

Implementation of secure sketch/fuzzy extractor like algorithms is first proposed by Juel in [12] called *fuzzy commitment scheme*, which employs hamming distance error correction to recover data from noises.

Later, *fuzzy vault scheme* is proposed by [11] with the concept of Locking set and Unlocking set. They add all points in original minutiae template and large amount of *other points* to the Locking set, such that all the points in original template satisfy a polynomial while the *other points* don't. Reed-Solomon decoding schemes is employed to work together with Unlocking set to recover template back from replacement noises.

[6] proposed *Fingerprint vault* scheme which brings in the idea of fuzzy vault scheme. It adds randomly generated chaffs to the original minutiae set to build a public sketch called fingerprint vault. It tries to confuse people by these chaff points since we are unable to distinguish original minutiae from chaff without the helps of any additional information. The fingerprint vault can be used to recover templates from white noises and Reed-Solomon[20] error correcting code is employed to handle the replacement noises.

A variance of MAC(message authentication code), AMAC(approximate message authentication code) is firstly proposed by Graveman[9] for robust image authentication. It produces MAC like binary code and is able to authenticate messages with permissive noises/alters.

1.2 Motivation

As we described in the previous section, fingerprint vault scheme adds random chaff points C to the original minutiae set X and publishes the sketch $P_X = X \cup C$. By the simple observation on P_X, a brute-force attacker who tries to find X out of P_X would have to enumerate all $|X|$-size subsets of P_X unless additional information is provided. Suppose $|X| = s$ and $|P_X| = m$, he needs on average $\frac{1}{2}\binom{m}{s}$ trials to find X. This complexity has been used to measure the security of fingerprint vault scheme. However, an interesting question would be: ***Is it possible to distinguish*** X ***from*** C***? or Is there a smart attacker who could perform better than brute-force attackers?***

In addition, with all those requirements introduced in the Section 1.1 for robust authentication on fingerprint templates, we find fingerprint vault is not ideal for being the authentication information, because firstly, we want to extract only a small MAC-size code for cryptographic usage; and secondly, the fingerprint vault always contains the correct information of the original minutiae. Later we are inspired by [9, 2, 21, 7], where the concept of *Approximate Message Authentication Code* and many of its variances are proposed for robust image authentication, e.g. 8192-bit difference in a 1 megabyte (2^{23} bit) image file results in an expected 7.07-bit difference in the output 128-bit AMAC code. If we could produce similar locality preserving hashes from 2-D point sets, we can use the hashes as keys for encryption or other cryptographic usage, and classic secure sketch on bits can be used to recover noises

on them. This leads to a second question: ***Would it be possible to design such an AMAC code for 2-D point sets, such as fingerprint templates?***

1.3 Problem Definitions

Motivated by two questions in 1.2, we formally define our problems in this section:

Finding original minutiae hidden among the chaff

We express fingerprint template as 2-D point set X, where each point $x_i \in X$ is a minutiae points extracted from a finger. Each point will have a pair of Euclidean coordinates (x, y) respected to a domain of size $[0, n] \times [0, n]$. The chaff set C will be generated randomly and the chaff point will be added to the domain one by one with constraints that no distance between any two points in the domain will be greater than δ. After we combined X and C together to get a sketch $P_X = X \cup C$, we observe a point set and we are not able to tell which point is an original minutia and which point is chaff point. Our first problem will be to find X out of P_X given only the information about P_X.

Design robust authentication function: *AMAC for 2-D point set*

We now consider more general cases, where the authentication targets are 2-D point-sets, such as fingerprint templates, handwritten images and other

biometric information. Our second problem is to design a method to extract information from these point sets and produce a robust authentication function: *AMAC for 2-D point set*. It should satisfy following properties:

- AMAC is able to authenticate a message to an identity
- AMAC ensures robustness and sensitivity with high probability
- AMAC outputs small fixed length binary code
- AMAC resists to preimage attacks.

1.4 Contributions

The two main contributions of this thesis are:

1. A smart attack model to fingerprint vault scheme is proposed, which finds original minutiae points hidden among the chaff 2192.7 times faster than a brute-force attacker. In extremely cases, we may have 10^7 times speed up compared with a brute-force attacker.[5]
2. We propose a general robust authentication function: *AMAC for 2-D point set templates*, such as fingerprint templates, watermarks, handwritten images and other image templates. This function can produce bit strings which could be used for further construction of secure sketches. A naive version of AMAC for 2-D point set templates can resist to pre-image attacks with complexity around 2^{81}.

Attack Model on Fingerprint Vault Scheme

We observe that a chaff point generated later tends to have smaller freedom space around it. We utilize this observation and experiment out the probability lookup table with freedom space and probability of being a minutiae as attributes. Then we can obtain the approximate likelihood for any $|X|$ size subset of P_X to be the original minutiae set. By querying a blackbox according to the likelihood, we achieved our speedup in finding minutiae compared with brute-force attackers.

We find if the number of minutiae $|X| = 1$ and the average vault size $|R| = 312.6$, using the brute-force search needs 156.5 queries, whereas using our method, we need only 100.0 queries on average. If $|X| = 38$, our method achieves a speedup of 2192.7 by choosing the `Lookup` appropriately.

Build AMAC for 2-D Point Sets Authentication with robustness

We propose using a transformation to map the point set to a high dimension Euclidean space. By doing so, we can apply known secure sketch technique to extract consistent secret bits for authentication.

We find if we draw a line l intersecting with the 2-D point set X, we will have some points on one side of this line, denoted as set L, while others on the other side, denoted as set R. The difference $|L| - |R|$ is considered as a feature value $FV_X(l)$ corresponding to l on X. If we have large number of such lines and large number of possible 2-D point sets, we could build a matrix $\mathcal{M}$ with each row as different $FV_X(l_i)$ for one point set X and different lines $l_1, l_2, ..., l_n$; and each column as the feature values $FV_{X_j}(l)$ for

a single line l with multiple possible underlaying point sets $X_1, X_2, ...X_s$. We then perform PCA transformation on $\mathcal{M}$ and obtain a set of basis B.

Given any 2-D point set template, we generate the AMAC code in the following way: 1. extract the information from the 2-D point set; 2. project it to B to get a projected vector V_B; 3. quantize V_B to a fixed length binary string such as 128 bits.

If we allows no more than 1% noises, the probability of finding a template mapping to a given AMAC code is only 1.10×10^{-24}. It means around 2^{81} trials are needed in order to find a point set mapping to a similar AMAC value.

1.5 Organization

In Chapter 2, we will introduce the related work in the authentication area, especially for robust authentication function suitable for fingerprint templates. The first main part of the thesis (Chapter 3,4,5) is to solve the problem of finding minutiae hidden from chaff. In Chapter 3, we model fingerprint vault scheme and describe point generation process; then in Chapter 4, we propose our attack model to find the minutiae hidden among the chaff and in Chapter 5, we support our findings by showing experiment results. The second main part of this thesis is to propose the method for extracting consistent bits from 2-D point sets. In Chapter 6, we give a design for AMAC for 2-D point set protocol; in Chapter 7, the effectiveness and boundaries of

the protocol are analyzed ; in Chapter 8, we present the experiments which reflects the properties of the AMAC code. In Chapter 9, we have conclusions and future works.

Chapter 2

Related Works

2.1 MAC and Hash Functions

A cryptographic message authentication code (MAC) is used to authenticate a message. A MAC algorithm usually uses hash functions to provide underlaying secrecy. Hash function is designed according to one-way, strong-collision resistance, weak-collision resistance criteria. The nice property for MAC code is it ensures the integrity of the message, but sometimes, it is too strict to have one message strictly mapping to a MAC since robustness is required for many types of messages, especially for messages involving the positioning. The information may be masked by noises, e.g. fingerprint templates, images with watermarks, and so on. Or else, the messages captured may not always be the same, yet they follow similar patterns from time to time, e.g. autographs, facial images, and so on. Other than this, we find

cryptographic MAC is more comfortable to deal with binary documents, but lacks of ability to process fingerprint templates, or more generally, 2-D point sets. The limitation of classic cryptographic urges us to invent a more appropriate approach to map this class of information to message authentication codes.

2.2 Secure Sketch, Fuzzy Extractor

Dodis [8] formally defined *secure sketch* and *fuzzy extractor* for turning biometric information into keys usable for any cryptographic application as well as reliably and securely authenticating biometric data. They also proposed 3 constructions: hamming distance, set difference and edit difference. The entropy loss for these constructions is about the same according to their analysis. BCH-based secure sketch[13], is constructed to solve set difference scenarios.

A *secure sketch* $SS(\cdot)$ is a deterministic function produces a random string $SS(w)$ for a uniform distributed biometric template w. It has the ability to recover the w given an input w' that is close enough to w by a recover function $Rec(w', SS(w)) = w$. The limitation of a secure sketch is it only accepts uniform template inputs.

A *fuzzy extractor* is similar to secure sketch but is able to handle inputs of non-uniform distributed biometric templates. It could generate uniformly random string R and P (which are stored for authentication) from biometric

input w. If an input $w^{'}$ is close enough to w, we could recover R by feeding $w^{'}$ and P to a reproduction function.

More specifically, we denote P_X as the secure sketch generated from a template X. P_X has the property that: given a template Y and a secure sketch P_X, we are able to recover X only if Y is close to X. The *closeness* here reflects the robustness we can tolerate.

2.3 Secure Sketch Implementations

Many secure sketch implementations try to do error correction on the noisy input data. Two main approaches of error correcting in constructing secure sketches are: 1. biometric templates are expressed in the vectors form where the distance could be measured using hamming distance; 2. biometric templates are abstracted to point sets, where the distance of two template is measured through set difference.

Fuzzy commitment scheme [12] is one of the earliest approach of handling error tolerance on noisy input using hamming distance.

Set difference is a new way of producing secure sketches, which is firstly mentioned in *fuzzy vault scheme* [11]. It uses Reed-Solomon algorithm[20] underlaying, and is able to correct up to $\frac{n-k}{2}$ errors, where n is the number of secrets and k is the order of a polynomial function.

In [6], *fingerprint vault* scheme is proposed to apply the *fuzzy vault* scheme

to fingerprint authentication. The fingerprint vault scheme consists of two parts: The first part is to extract a public point set $R = (X \cup C)$, where X is the set containing original minutiae points and C is a set of random chosen chaff points added to X. The whole set R is δ-separated, where any two points in R are apart by at least δ distance. The points in C are selected one by one following follow process: 1. select a random point in 2-D template space, 2. if this point is within δ distance of any points in X or any selected chaff points, we discard it, otherwise, we add it into C. The selection process would be repeated until no more chaff points could be added or sufficient amount of points have been selected. The second part is to do authentication with robustness.

Recently, Chang[4] proposed small secure sketch for point set difference, where set reconciliation[15] technique is used.

However, all of these schemes don't produce small fixed length secure sketch. They either keep the whole encoded information or add more redundant information to confuse people. Whereas in authentication, we are more comfortable to keep only small compressed trunks of binary strings. Further more, many biometric point sets are not suitable for being projected to Euclidean space directly.

Shielding function is an implementation of fuzzy extractor which works on continues domain. It relies on δ-contracting,ϵ-revealing function. It uses challenge and response approach with the involvement of public database.[14] However, the involvement of a third party makes the problem complicated.

2.4 AMAC AIAC and AIMAC

Another totally different approach is the construction of *approximate message authentication codes*[9, 2, 21, 7] on robust image authentications, where images that are corrupted by certain levels of noises can be authenticated successfully by the AMAC code.

The first version of AMAC[9] consists of three steps: *Initialization*, *Formatting and Randomization*, *Two Rounds of Majority Calculation*.

- *Initialization* The key K and initial vector I are selected and they are used to seed the PRNG.
- *Formatting and Randomization* The message M is padded with zeros to the length of $L \times R \times S$ and then chopped into L columns with $R \times S$ rows each. The result matrix would be masked by a new set of pseudo-random bits generated by P, which could be denoted as T_0.
- *Two Rounds of Majority Calculation* First round is to take R rows from T_0 at a time to form S subarrays. For each of R rows and L columns, denote sub-arrays as T_0^0, $T_0^1 ... T_0^{s-1}$. For each T_0^k, we compute the majority bits of that column. Hence a new matrix T of $S \times L$ is formed. Second round is to carry out majority of each column of T to obtain L bits. These L is output as the AMAC.

The intermediate steps look very similar to the cryptographic hash function design, such as MD5 and SHA-1. However, this scheme doesn't capture

the positioning information between the objects/pixels in the images.

Later, Xie[21] proposed Approximate Image Message Authentication Codes to address the above problem. In stead of processing direct on rows and columns, she divides the image into blocks to preserve the locality information. She also introduces a *guarding zone* in the image's histogram by transforming it and creating a gap around the threshold. However, the result is mainly empirical and is lack of reasoning.

Other attempts are studied to refine the extraction of biometric features so that the features are invariant to permissible noises[22]. However, such system doesn't have high reliability.

2.5 Principle Components Analysis

Principal components analysis (PCA) [10] simplifies a dataset using linear transformation. The linear transformation chooses a new coordinate system for the dataset such that first few coordinates in the new system reflect the most important features of the dataset. PCA is also called the Karhunen-Loeve transform or the Hotelling transform. PCA provides optimal linear transformation for keeping the subspace that has largest variance. Unlike other linear transformation, the PCA does not have a fixed set of basis vectors. Its basis vectors depend on the data set which could be trained with actual possible inputs.

Chapter 3

Model the Fingerprint Vault Scheme and Brute-force Attacker

3.1 General Approach

We study the fingerprint vault's chaff points generation process which is modeled as *online parking process*[17]. Notice that the chaff points are generated one by one in a randomly and uniformly manner. We observe that a chaff point that is generated later tends to have smaller freedom space around it. We would define this freedom as *free area* formally later. Based on this observation, we perform extensive simulations and indeed verify the different location arrangement between points would affect a point's likelihood to be

a minutia. We also admit that if all the points in the vault are purposely distributed with equal/similar distance, we will have difficulty in applying our method to distinguish the original points from chaff.

Based on *online parking process*, we build a `Lookup` table through experiments, where the first column indicates the free area and the second column is the corresponding probability for this point to be a minutiae. Hence, given the number of minutiae k in a fingerprint vault, we could calculate the probability likelihood for any k points to be the original minutiae set. Starting from the highest probability to the lowest probability, we check whether the k points are the original minutiae set by querying a blackbox. We use the number of queries we send as the measurement to compare with a brute-force attacker. The ratio between two queries, namely, the brute-force way's query number over our way's query number is obtained from experiments.

3.2 Notations

X, Y	: X is the set of the original minutiae. Y is a noisy version of X.
s	: Number of minutiae. $s = \|X\|$.
C, R, m	: C is the set of chaff. $R = X \cup C$ and $m = \|R\|$.
n	: Width of the domain. All points are in $[0, n] \times [0, n]$.

P_X	: P_X is the sketch generated from X.
$\mathcal{A}(W)$	: Available region of a point set W.
$\mathcal{F}_{\widetilde{R}}(x), \mathcal{F}(x)$	: Free area of x in the point set $\widetilde{R}$.
`LookUp`	: Look up table used by the attackers.
A_x	: The arrival order of x, given that x is selected.

3.3 Online Parking Process

The online parking problem is initially introduced from such a scenario: We have a single line car park which is represented as an interval $[0, x]$, x is a large positive number. Each car (with unit width) arrives at the car park follows Poisson process and it chooses a number y from $[0, x-1]$ uniformly and randomly. If the interval $[y, y+1]$ is empty, the car parks in that slot, and if the interval overlaps with some previous arrived cars, it tries the process again, because two cars are not allowed to park with overlapping. Cars keep on arriving until it cannot find any slots which is able to fit them. The mean value of cars can fit in this interval is called *Rényi*'s Parking Constants[17].

In 2-D, we selects a set of points one-by-one uniformly and randomly from the domain $[0, n] \times [0, n]$ as in 1-D. Since it is in 2-D, we consider the radius 0.5 circle centered at each point as a *car*. If a point is within unit distance from any previously selected points, which means the car overlaps with a previous car, it is discarded. If not, it is selected. The process is repeated

until the stopping condition is met. We refer this way of point generation process as *Online Parking Process*. Two stopping conditions are defined as follows:

1. The 2-D parking process is repeated until no more points can be selected.

2. The 2-D parking process is repeated until a predetermined number of points are selected.

We will use the first termination condition to generate the sketch P_X. We also studies the effects of the second condition in Section 5.6.

For each selected point, if it is the k-th point selected, then we say that its *arrival order* is k.

3.4 Fingerprint Vault Generation Process

A fingerprint vault, which is referred as a sketch $P_X = X \cup C$, consists of two parts: X as a set of minutiae points and C as a set of chaff. We first generate X and then generate C.

We define the minutiae set X as a set of s points chosen from domain $[0, n] \times [0, n]$. The set X is δ-separated, where $\delta = 1$. We use X to recover white noise using its δ-separated property. Although in reality the minutiae might follow some distribution, we only consider the more general case, where

the minutiae points are also obtained from online parking process mentioned in the previous section. We feel that with the knowledge of minutiae distribution, we are able to distinguish minutiae from chaff better, and we are able to design better `Lookup` functions which increase our speedup compared with brute-force attackers even more.

We define the chaff set C as a set of points generated one by one using the same online parking process. The difference between the C and X in our model is that, X contains all the points whose arrival orders are less than or equal to $|X| = s$, but C contains all the points whose arrival orders are larger than s. The set C is used to recover replacement noise. For example, Juels et al. [11] proposed using a polynomial of degree $(s - 2t + 1)$, and employed RS in decoding. t is the number of replacement noises can be corrected by the sketch. All the points in X satisfy the polynomial while all the points in C don't.

The fingerprint vault P_X reveals some level of information of X, since a minutia point in X must be one of the point in P_X. Further more, in reality, since the sketch tolerates some mount of replacement noises, using less than $|X| = s$ points, we might be able to pass the authentication.

3.5 Attacking Model

The goal of attackers is to find a subset Y with size $|X|$ which satisfies $Y = X$. The attackers are allowed to present such a Y set to a blackbox

for confirmation. We define the effectiveness of the attack as the number of queries an attacker would send to this blackbox. The blackbox is a function which takes in a set of points Y, and replies `YES` if $Y = X$, where X is the original minutiae set embedded in the fingerprint vault P_X, or replies `NO` otherwise. In practice, a blackbox can be designed to extract a key from the input point set and use this key to perform an encryption on a file. If the encrypted file matches a system stored version, the blackbox outputs `YES`, otherwise outputs `NO`.

In reality, an attacker may try to send smart queries to the blackbox with the helps from the techniques such as Reed Solomon Code or BCH code. However, in this model, we assume attackers do not have the knowledge about these error correcting codes. Further more, we assume the offline calculation is also not counted in measuring the effectiveness of the attacks. It is more appropriate and convenient to count only the blackbox calls since usually, a black box is a remote server which attackers may only have limited access to and attackers may have huge local computational powers.

3.6 Brute-force attacker

We call a set Y a *candidate* of a given sketch(fingerprint vault), if Y is a subset of P_X and $|Y| = |X|$. A brute-force attacker will extract a candidate Y from P_X and sends Y to the blackbox we defined earlier. He will query the blackbox using all possible candidates consistent with the given sketch until a `YES` is obtained. Since the size $|X| = s$ subsets of a sketch P_X is $\binom{m}{s}$

when $|X \cup C| = m$. They are the candidates a brute-force attacker may send to the blackbox.

Notice the replacement sketch can also reduce the number of possible candidates. If the replacement sketch can correct up to t errors, and the set-difference scheme [11] is employed, then the average number of candidate consistent with both white noise and replacement sketch is approximately

$$\binom{m}{s} m^{-(s-2t)}.$$

We assume that the brute-force attacker is lack of knowledge about what error correcting code the blackbox is using. Hence, the expected number of queries required by a brute-force attacker is half of the total number of candidates.

3.7 Definition of Free area

Definition 1. *Given a set of points W, define $\mathcal{A}(W)$, the* available region*, to be the set*

$$\mathcal{A}(W) = \{x \in [0, n] \times [0, n] : \textit{ for all } w \in W, \|x - w\|_2 > 1\}.$$

We can add a point in the available region to W if and only if after adding the point, W still remains separated with the condition that $\|w_1 - w_2\|_2 > 1$ for any $w_1, w_2 \in W$.

Definition 2. *For a point set $\widetilde{R}$ and a point $x \in \widetilde{R}$, define the* free area *of x with respect to $\widetilde{R}$ as,*

$$\mathcal{F}_{\widetilde{R}}(x) = |\mathcal{A}(\widetilde{R} - \{x\}) - \mathcal{A}(\widetilde{R})|, \tag{3.1}$$

where "$-$" is the set difference operator, and $|\ \ |$ gives the area of the region.

Figure 3.1 illustrates the free area. For convenience reason, we omit $\widetilde{R}$ and write the free area as $\mathcal{F}(x)$.

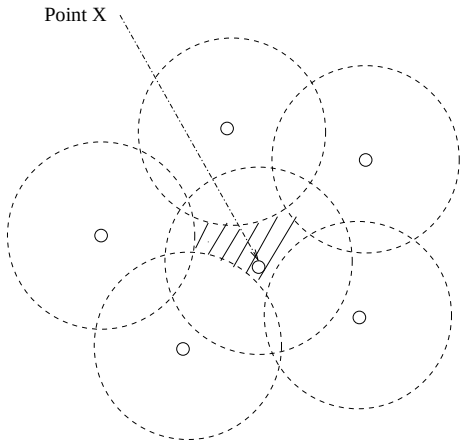

Figure 3.1: Illustration of free area. $\mathcal{F}_R(x)$ is the area of the shadow. Each circle is a unit disk.

Consider the online parking process. Let A_x be the random variable on the arrival order of x, given that x is selected. Let us write $\mathcal{F}(x) = f$ as the event that x is selected and contained in some point set, and its free area in that point set is f. We are interested in the conditional probability

$$Pr(A_x \leq s \mid \quad \mathcal{F}(x) = f). \tag{3.2}$$

Since X and C follow the distribution of online parking, we can treat R as the output of an online parking process. Hence, if the arrival order of x is not more than s, x is a minutia. Although the attackers know the point set R, the conditional probability (3.2) doesn't explore the full knowledge of R. Instead, only the free area of x is used. Nevertheless, such partial information is sufficient for us to distinguishing the minutiae from chaff.

3.8 Differences from Clancy et al. method.

The model we designed has two differences from the original fingerprint vault model proposed by Clancy et al. [6]. Firstly, we measure the effectiveness of an attacker by the number of queries to the blackbox. Whereas Clancy et al. considered the computational complexities required by the attacker, and the authentic user during decoding process(with respect to a specific decoding algorithm). The effectiveness of an attacker is defined as the ratio of the steps taken by the attacker over the authentic user. Secondly, Clancy et al. stopped generating more points when a predetermined number of sketch points is reached so that the authentic user can decode efficiently. We assume the attacker has strong computational power and do not consider the decoding complexity and hence generate as many chaff points as possible.

Chapter 4

Attacks to Fingerprint Vault Based on Conditional Probability

Our main observation is that, for $f_0 > f_1$ and any s,

$$\Pr(\ A_x \leq s \mid\ \mathcal{F}(x) = f_0) > \Pr(\ A_x \leq s \mid\ \mathcal{F}(x) = f_1). \tag{4.1}$$

That is, for a R randomly generated by online parking, if a point $x \in R$ has larger free area compared to another point y in R, it is more likely that x arrived earlier than y. This is intuitive when we have an empty space, if now we throw only one disk into the space, it can be located at anywhere within the boundary. However, after we already have such a disk in the space, we throw in another disk, provided it cannot overlap with the previous disk, we

have less freedom or fewer choices to put it.

Unfortunately, we are unable to analytically prove the observation because many of the related problems are remained open since $1950s$ [16]. Nevertheless, we have extensive simulations to support the claim and observation. Figure 5.2 shows an estimation of the likelihood function. Note that each function is increasing with respect to the free area.

4.1 Identifying one point, $s = 1$

We start from simple case $s = 1$, that is, there is only one minutia point in the fingerprint vault, $X = 1$. Suppose we have obtained a sketch P_X, we have all the candidates as singleton subsets of P_X. Recall all the minutiae and chaff are both generated using the online parking process. In this scenario, an attacker is to find the very first point that arrives in the point generation process. Suppose a sketch P_X has m points, then a brute-force attacker needs to send $m/2$ queries to the blackbox on average.

Instead of randomly choosing a candidate to query the blackbox, our attacker carries out the following steps:

1. The attacker computes $\mathcal{F}(x)$ for all $x \in P_X$.

2. Next, it lists down all the points in P_X in decreasing order with respect to $\mathcal{F}(x)$ value. Since we believe a point that arrives later in the online parking process tends to have small $\mathcal{F}(x)$ value, we send the points in

this order one by one to the blackbox until a `YES` is returned.

4.2 Likelihood of the first s points

In online parking process, we notice the later generated points are restricted by the previous generated points. Hence the online parking process is not memoryless and there is dependency between two points. Particularly, $\Pr(A_x < s) > \Pr(A_x < s | A_y < s)$ since if a previously point has arrival order less than s, the later coming points will have less chance to be the first s points.

Nevertheless, with the assumption that x and y are not close to each other, the effect of one point on the other should not be significant. Hence, we employ the following approximation:

$$\begin{aligned} &\Pr(A_x \leq s, A_y \leq s \mid \ \mathcal{F}(x) = f_1, \mathcal{F}(y) = f_2) \approx \\ &\quad \Pr(A_x \leq s \mid \ \mathcal{F}(x) = f_1) \cdot \Pr(A_y \leq s \mid \ \mathcal{F}(y) = f_2). \end{aligned} \tag{4.2}$$

Using (4.2), we can obtain an approximation of the likelihood for each candidate consistent to P_X. This leads to the following attacker:

1. Computes the likelihood of each candidate consistent to P_X.
2. Enumerates the candidates in decreasing order with respect to their likelihood. Next, send the enumerated candidates to the blackbox until the blackbox outputs `YES`.

For example, if the candidate is $\{x_1, x_2, ..., x_s\}$, we denote the value $\prod_{i=1}^{s} \texttt{LookUp}(\mathcal{F}(x_i))$ as the likelihood of each candidate, where `LookUp` is a predeterminate function using simulation.

We assume if an attacker has sent a few candidates to the blackbox and they are all not the correct original, the attacker will not use this information to choose the next candidate. However, in the reality, an attacker may immediately discard a candidate set of points if he tries several times with small modifications on that candidate.

In our simulation, we experiment with various ways to estimate the likelihood. Since in $\prod_{i=1}^{s} \texttt{LookUp}(\mathcal{F}(x_i))$, we use the estimation 4.2, which in fact underestimates the likelihood of a candidate to be X. Hence we try to use identity function as lookup, that is, $\texttt{LookUp}(i) = i$, this actually increase the likelihood to a more accurate value. We find the identity lookup function can achieve noticeable speedup over the brute-force-attacker.

4.3 Min-entropy retained by publishing the sketch

Measuring the remaining entropy is a way to judge the security of a secure sketch. The sketch P_X is made public, hence there is information leakage on the original minutiae point set. We want the remaining entropy to stay high so that it is less risky to publish the fingerprint vault. For convenience reason, Dodis et al. [8] proposed the concept of min-entropy of A given B,

which is,

$$\widetilde{H}_\infty(A|B) = \ -\log(\mathbf{E}_{b\leftarrow B}[\max_a \Pr(A=a|B=b)]). \tag{4.3}$$

We denote X and P_X as the random variables for the minutiae and the sketch respectively, the min-entropy loss due to the sketch is defined as,

$$H_\infty(X) - \widetilde{H}_\infty(X|P_X),$$

where the min-entropy $H_\infty(X) = -\log\ (\max_a \Pr(X=a))$.

Online parking is not easy to analyze due to the reason that many basic properties are mathematically open problems. Hence the bound on the entropy loss also seems to be difficult to obtain. We take an alternative and estimate the entropy loss by simulations and the approximation (4.2), we can obtain an estimation of

$$\max_{\{x_1,x_2,\ldots,x_s\}} \Pr(X=\{x_1,\ldots,x_s\}|\mathcal{F}_R(x_1),\ldots,\mathcal{F}_R(x_s)). \tag{4.4}$$

Note that for random variables A and B, and a deterministic function f,

$$\max_a \Pr(A=a|B=b) \geq \max_a \Pr(A=a|f(B)=f(b)).$$

Therefore, by substituting A with X, B with P_X and f with $\mathcal{F}$, we have $\max_a \Pr(X=a|P_X=R)$ is greater or equal to the likelihood in (4.4). We use this to estimate an upper bound on the min-entropy, which in turn gives

us an lower bound on entropy loss. In the later experiments, we measure this value empirically.

Chapter 5

Experiments on Attacking Fingerprint Vault Scheme

5.1 Experiment Settings.

For convenience, we simulate the online parking process on the discretized domain $[0, n] \times [0, n]$. Therefore, minutiae and chaff points are selected from a set of discrete points. Each unit interval is discretized into 100 points. Hence, there are 100^2 points in $[0, 1) \times [0, 1)$. For each experiment, we collected 10000 samples. Each sample is a point set R obtained through online parking. For each point in R, its arrival order is recorded and its free area is approximated by counting the discrete points in the area.

The experiments are conducted in $[0, n] \times [0, n]$ for $n = 50$ and $n = 22$.

The average $|R|$ is 1668.4 and 312.6 for $n = 50$ and $n = 22$ respectively. By treating each point as a disk of diameter 1, the average packing density (for both $n = 50$ and $n = 22$) is about 0.525. This is slightly less than the Palasti's conjecture[16] of 0.559, probably due to the different treatment of the domain boundary.

We also conducted experiments in 1-D. That is, the domain is the interval $[0, n]$, and for any two selected points x and y, $|x - y| \geq 1$. The free area of a 1-D point x can be defined similarly as in (3.1). In 1D, the free area of x is simply $(x_{\mathtt{r}} - x_{\mathtt{l}} - 2)$ where $x_{\mathtt{r}}$ and $x_{\mathtt{l}}$ is the right and left neighbors of x respectively. For $n = 1340$, the average $|R|$ is 994.8, which gives packing density of 0.743.

5.2 Likelihood

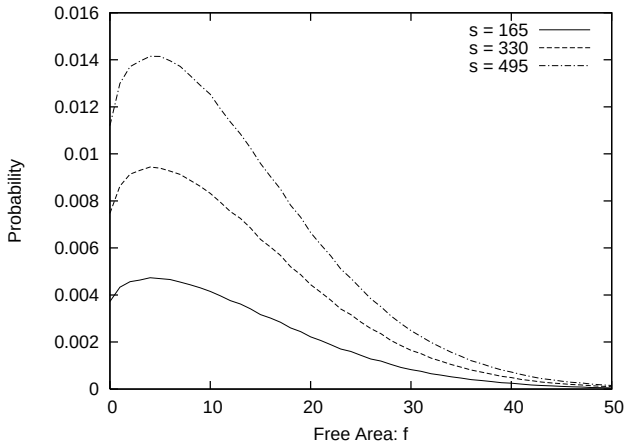

Figure 5.1: Probability density function, $Pr(A_x \leq s \cap F(x) = f)$, for $s = 165, 300$ and 495. The domain is $[0, n] \times [0, n]$ where $n = 50$.

From the 10000 samples, we can estimate the conditional probability $\Pr(A_x \leq s \mid \mathcal{F}(x) = f)$ by first estimating $\Pr((A_x \leq s) \cap (\mathcal{F}(x) = f))$ and $\Pr(\mathcal{F}(x) = f)$. Figure 5.1 plots $\Pr((A_s \leq s) \cap (\mathcal{F}(x) = f))$ against the free area f for different s. A function in Figure 5.2 shows $\Pr(F(x) = f)$ with respect to f. Observe in Figure 5.2 that a large proportion of points have small free space. In contrast, as illustrated in Figure 5.1, for points that arrive early, relatively small proportion of them have small free space. This implies that, given that a point has large free area, it is more likely to have arrived early, and hence more likely to be a minutia.

Figure 5.2 shows the likelihood function for different s. Note that $\Pr(A_x \leq s \mid \mathcal{F}(x) = f)$ is increasing as a function of f. Since the average $|R|$ is 1668.4, the probability that a randomly chosen point from R to arrive not later than 165 is about 0.1. From the graph, the likelihood $\Pr(A_x \leq 165 \mid \mathcal{F}(x) = 50)$ is more than 0.15. Hence a point with free area more than 50 is 1.5 times as likely to arrive not later than 165, compared to a randomly chosen point.

Similar observations can be made for experiments in the 1D domain, illustrated in Figure 5.3.

For different n and s, it seems that the conditional probabilities are almost the same as long as the ratio (s/n^d) is the same, where d is the dimension of the domain. This is illustrated in Figure 5.4 where $d = 2$.

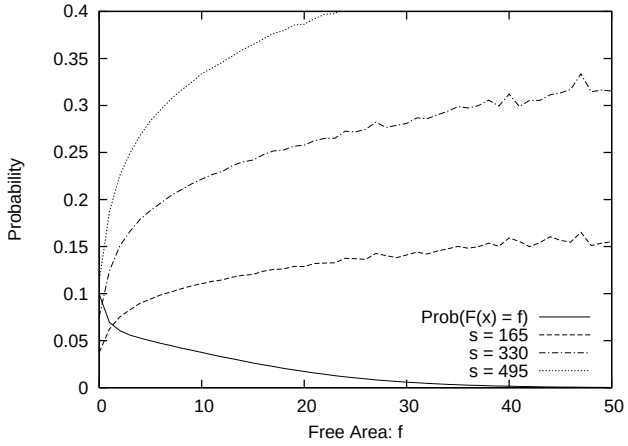

Figure 5.2: Distribution of free area, $\Pr(F(x) = f)$, and conditional probability of arrival order given free area, $\Pr(A_x < s | F(x) = f)$, for different $s = 165, 330, 495$. The domain is $[0, n] \times [0, n]$ where $n = 50$. The average $|R| = 1668.4$.

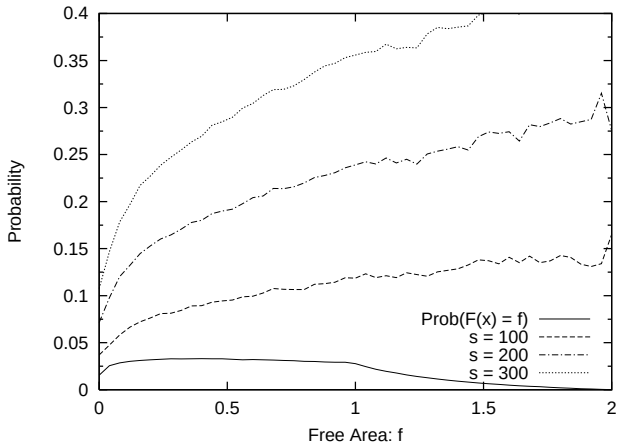

Figure 5.3: Distribution of free area, $\Pr(F(x) = f)$, and conditional probability of arrival order given free area, $\Pr(A_x < s | F(x) = f)$, for different $s = 100, 200, 300$. The domain is $[0, n]$ where $n = 1340$. The average $|R| = 994.8$.

5.3 Brute-force-attacker for $s = 1$

When $|X| = 1$, our attacker simply computes `LookUp`(x) for all $x \in R$, and sends them to the blackbox according to the looked-up values in descending

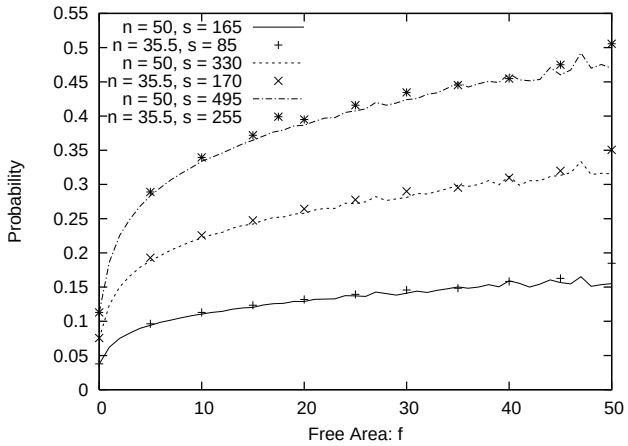

Figure 5.4: Comparison of $\Pr(A_x < s | F(x) = f)$ with different n and s. There are 3 pairs of functions. For each pair, the ratio (s/n^d) is approximately the same. For example, $(165/50^2) \approx (85/35.5^2)$.

order. For each sample R, the number of blackbox calls required is the number of points in R whose looked-up value is larger or equal to the looked-up value of the sole minutia. Figure 5.5 gives the histogram of the number of calls, and the average is 100. Note that the brute-force attacker on average takes 156.5 calls.

5.4 Brute-force attacker for $s > 1$

The average speedup factor comparing to the brute-force attacker can be estimated in the following ways. Consider a sample, which is a point set $R = \{x_1, x_2, \ldots, x_m\}$. We can compute the likelihood of the actual minutiae X. Recall that we estimate the likelihood of the set X by $\prod_{x \in X} \texttt{LookUp}(x)$. Also recall that our attacker sends the candidates to the blackbox in the

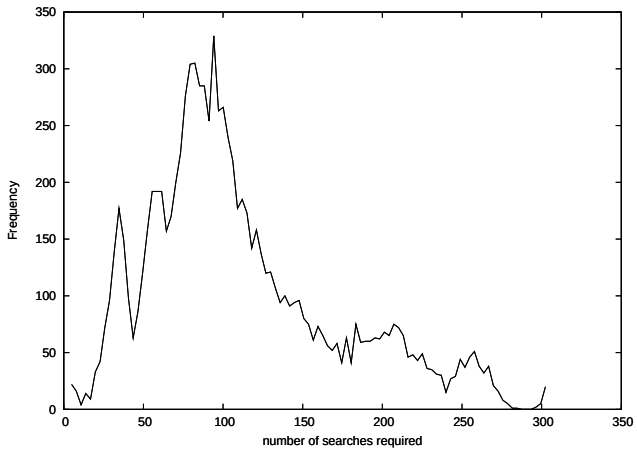

Figure 5.5: Histogram of number of calls made by our attacker. Number of Bins= 100, Bin Size= 2.97

order of decreasing likelihood. Thus, the number of blackbox calls required to hit X is same as the number of candidates whose likelihood is larger than that of X.

Now, consider the set $L = \{\log(\texttt{LookUp}(x_1)), \ldots,$ $\log(\texttt{LookUp}(x_m))\}$. By Central Limit Theorem, the distribution of the sum of s randomly chosen numbers from L can be approximated by a normal distribution, whose mean and variance can be derived from the mean and variance of L. From this normal distribution, we can estimate the proportion of subsets of R whose likelihood is larger than that of X. Assuming that the candidates are randomly located among all subsets of R, we can obtain the speedup factor provided by the attacker.

Figure 5.6 shows the histogram of the speedup factor, when $s = 38$ and expected number of points in R is 312.6. The average speedup for the 10000 samples is 77.5, and the geometric mean is 18.0. It also shows the result

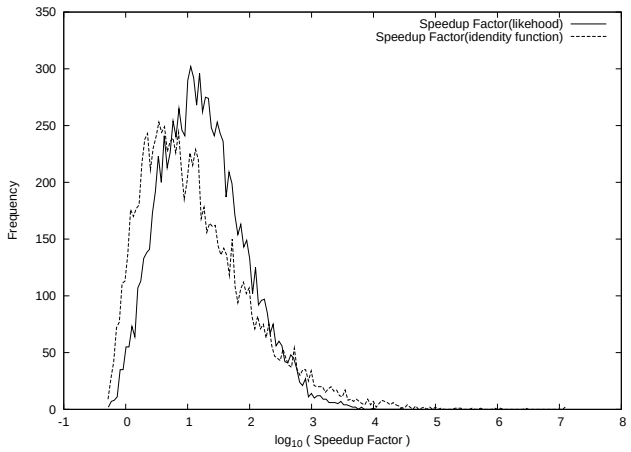

Figure 5.6: Comparison on speedup factor. Using the likelihood or identity function as the lookup table.

for the same s, but using a different `LookUp`. Here, we simply choose the identity function as the lookup function. Interestingly, the average improves to 2192.7, and the geometric mean reduces to 13.88. Observe that for some samples, the speedup factor reaches 10^7. This is undesirable in security applications because it indicates that, with some probability, small but noticeable, the attack can be very successful.

5.5 Entropy loss

We want to estimate the min-entropy of the minutiae set X given the white noise sketch R, that is, $\widetilde{H}_\infty(X|R)$. Note that each sample R is a randomly chosen white noise sketch. Now, using the approximation in (4.2), the set $\{x_1, x_2, \ldots, x_s\}$ that maximizes the conditional probability $\Pr(X = \{x_1, \ldots, x_s\}|R)$ is the set with s largest looked-up value. Hence, for a sample R,

we can obtain $\max_a \Pr(X = a|R)$. By averaging over all samples, we have a lower bound of min-entropy of X given the white noise sketch. When $s = 38$, the min-entropy is at most 61.2 bits. In other words, by making the white noise sketch public, the min-entropy of the minutiae is reduced to at most 61.2.

The above estimate does not consider the replacement sketch. The entropy loss of many set-difference schemes is known. For example, if we employ the scheme by Juels et al. [11], and the replacement noise is $t = 3$, then the entropy loss due to the replacement sketch is at least $2t \log_2 |R| < 49.72$. As an approximation, let us assume that the replacement sketch is generated independently from the white noise sketch. Then, the min-entropy given the sketch P_X is at most $61.2 - 49.72 = 11.48$. From this value, we find the fingerprint vault scheme is not ideal since the sketch reveals too much information.

5.6 Online Parking with Fixed Number of Chaff

In previous sections, we employ the first stopping condition for online parking process (that is, the process stops when it is impossible to add any more chaff points). It would be interesting to investigate the second stopping condition which stops at a fixed number of chaff points. We conduct experiments with $m = 250$, 200, and 150, where m is the total number of chaff points. The

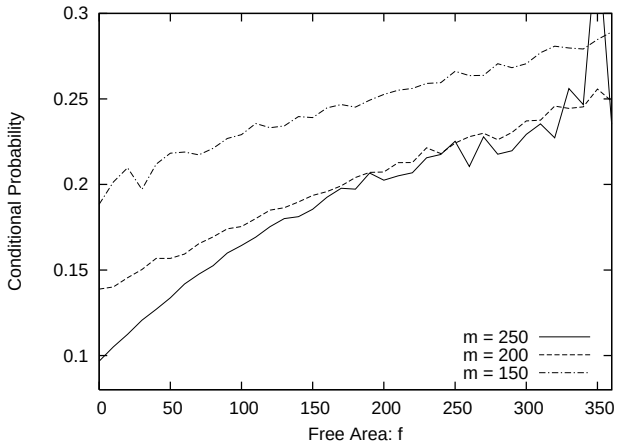

Figure 5.7: Conditional probability whereby the online parking generates fixed number of chaff points. The figure shows $\Pr(A_x < s|F(x) = f)$ where m, the number of chaff points, is 250, 200,or 150, and $s = 38$. The domain is $[0, n] \times [0, n]$, where n is 10.7.

result is illustrated in Figure 5.7. Observe that the conditional probability is still increasing although for smaller m, it increases more gradually.

Chapter 6

Design AMAC for 2-D Point Set Templates

6.1 General Approach

Our AMAC for 2-D point set protocol consists of three functions: AMAC trainer $f_{trainer}$, AMAC generator $f_{generator}$ and AMAC verifier $f_{verifier}$.

$f_{trainer}$ is a function which trains the AMAC protocol using a large number of real sample templates using principle components analysis (PCA). It takes in templates with secret key - a set of feature lines, and outputs a projection basis which serves as key of AMAC for 2D point set together with the set of feature lines.

$f_{generator}$ is a function which generates the AMAC codes. It takes in a

template and outputs a fixed length binary code using keys generated in $f_{trainer}$ process.

$f_{verifier}$ is a function which verify whether a given template matches an AMAC code. It takes in a template with an AMAC code, and outputs `YES` or `NO` based on whether the given template could map to an AMAC close enough to the given AMAC code.

6.2 The Model of 2-D Point Set Templates

Definition 3. *A* template *is a 2 dimension abstract point sets extracted from fingerprint minutiae sets, images or other documents.*

The abstract points of a template could be minutiae points in the fingerprint context, or pixels whose characteristic values are above certain threshold, in an image context. The template is defined in 2 dimension space because we intend to preserve positioning information of each abstract point from 2-D biometric message. Hence each point would have a pair of x-y coordinates indicating its position. We define the template to be a domain of $[0, n] \times [0, n]$ unless stating specifically.

Definition 4. *A* feature line l *is a line intersecting with a template which divides a template into two parts geographically. We denote the line function of* l*:* $ax + b + cy = 0$*, or more generally,* $f_l(x, y) = ax + b + cy$*, where* $f_l(x, y) = 0$.

In this thesis, a feature line is generated as by following ways: firstly, we uniformly and randomly pick a point p in the template domain; secondly, a degree θ between the line and the horizontal x-axis is chosen uniformly from $[0, \pi)$. This two value could determine a unique line that intersects the template.

Definition 5. *A* feature value *of a feature line l on template X is defined as $FV_X(l) = |A_X| - |B_X|$, where $A_X = \{(x, y)|(x, y) \in X \wedge f_l(x, y) < 0\}$ and $B_X = \{(x, y)|(x, y) \in X \wedge f_l(x, y) \geq 0\}$. In the cases with no confusions, we omit the X and denote the feature value associate to l as $FV(l)$.*

Definition 6. *A* feature vector *is of the form $v_X = (v_1, v_2, ..., v_m)$, where $v_i = FV_X(l_i)$, l_i are feature lines from $i = 1, ..., m$.*

Definition 7. *A robust hash function on point sets $\mathcal{M}$ is a function* `RH`*: $\mathcal{M} \rightarrow \{0, 1\}^k$ such that both* sensitivity *and* robustness *are guaranteed:*

1. Robustness : *if m_1, $m_2 \in \mathcal{M}$ and $dist_1(m_1, m_2) < \delta_1$ for some small δ_1, we have $dist_2(\texttt{RH}(m_1), \texttt{RH}(m_2)) < d_1$ for some small d_1.*

2. Sensitivity : *if m_1, $m_2 \in \mathcal{M}$ and $dist_1(m_1, m_2) > \delta_2$ for some small δ_2, we have $dist_2(\texttt{RH}(m_1), \texttt{RH}(m_2)) > d_2$ for some d_2.*

where $\delta_1 < \delta_2$, $d_1 < d_2$, $dist_1(,)$ is a function measures the difference between two point set templates and $dist_2(,)$ is a function measures the difference between two hashes.

We will relax the sensitivity rule in the definition such that, if $dist_1(m_1, m_2) >$

δ_2, there is very large probability (very close to 1) for $dist_2(\mathtt{RH}(m_1), \mathtt{RH}(m_2)) > d_2$.

Definition 8. *An approximate message authentication code for 2-D point set is a robustness hash function on 2-D point sets with relaxed sensitivity rule.*

Definition 9. *We model our noises occurs in a 2-D point set template as either of the following cases:*

1. *Some points in old template perturb to new locations nearby in the new template.*
2. *Some original points may be missing in the new template*
3. *Some new points may appear in the new template.*

In fingerprint template context, the first type of noise is mentioned as *white noise*, and the second and third types of noise is mentioned as *replacement noise*, which we have introduced in Chapter 1, See Figure 1.1. We assume for the first kind of noise, the perturb distance is not larger than γ, otherwise the noise will be treated as we firstly lose the original point and then introduce a new point.

In the next section, we formally define our robust hash function: approximate message authentication code for 2-D point set templates.

6.3 Approximate Message Authentication Code for 2-D Point Set Templates

Fingerprint templates are minutiae with x-y coordinates, which forms 2-D point-sets. The positioning between the minutiae, as well as the number of minutiae are the key information of a fingerprint template. In order to capture the different positions of minutiae, we use the secret key mentioned previously- the feature line sets. Since every line divides a template into two parts, the points in a template could be on either sides of a line. With the help of many such lines, the information about the positions of points in a template could be captured uniquely. For other images templates, we could set the similar approach, except the definition of *points* is change to pixels or pixel with characteristic value higher than a threshold.

Our AMAC for 2-D point set proposal consists of three functions: AMAC trainer $f_{trainer}$, AMAC generator $f_{generator}$ and AMAC verifier $f_{verifier}$, as Table 6.3, Table 6.3 and Table 6.3 shown. We first obtain the AMAC keys from $f_{trainer}$. Then we could produce AMAC code for any 2-D point set template. Finally, we are able to verify whether a template maps to a given AMAC using $f_{verifier}$.

Function name: $f_{trainer}$
Input: large number of sample templates
Output: a projection basis u
Key: a set of secret feature lines
For each template
feature vector = Feature Vector Generation (template, feature lines)
I = feature vector matrix
u = basis generation using PCA
output u

Table 6.1: Pseudo-code for Function $f_{trainer}$

Function name: $f_{generator}$
Input: a template
Output: a fixed length (e.g 128 bit) binary AMAC code
Key: a set of secret feature lines, a projection basis u
feature vector = Feature Vector Generation (template, feature lines)
projected vector = Project to basis u (feature vector)
AMAC = Map projected vector to AMAC (projected vector)
output AMAC

Table 6.2: Pseudo-code for Function $f_{generator}$

Function name: $f_{verifier}$
Input: a template, an AMAC
Output: `YES` or `No`
Key: a set of secret feature lines, a projection basis u
AMAC'= $f_{generator}$(template);
Distance = Number of Different Bits(AMAC', AMAC);
if (Distance < threshold)
output `YES`
else
output `NO`

Table 6.3: Pseudo-code for Function $f_{verifier}$

6.4 Feature Vectors Generation

For a given set of feature lines $\{l_1, l_2, ..., l_n\}$, we generate feature vector $v_{X_i} = (FV_{X_i}(l_1), FV_{X_i}(l_2), ..., FV_{X_i}(l_n))$ for any given template X_i.

In case we are in $f_{trainer}$ function, we need to use many possible X_i to build a feature vector matrix for future processing by PCA.

In case we are in $f_{generator}$, v_{X_i} will be used later for being projected to a projection basis obtained in $f_{trainer}$.

6.5 Obtain Projection Basis using PCA

PCA[10] process trains the AMAC generator using large samples of possible input biometric templates. PCA stands for Principle Components Analysis, which is a technique that can be used to simplify a dataset. It reduces the dimensions of the original dataset while retains those characteristics that contribute to the dataset most.

PCA first takes large number of feature vectors obtained in the previous steps to form a matrix $I_{n\times m}$ of size $n \times m$, where each row represents a feature vector on a template, and each column represents the possible feature values associate to a specific feature line. Notice the column vector of I may be correlated as shown in Figure 6.1. Line l_1 and l_2 are symmetric, the feature values of these two feature lines tends to satisfy the relationship $FV(l_1) \approx FV(l_2)$ if the underlaying point set follows Poisson Arrival model.

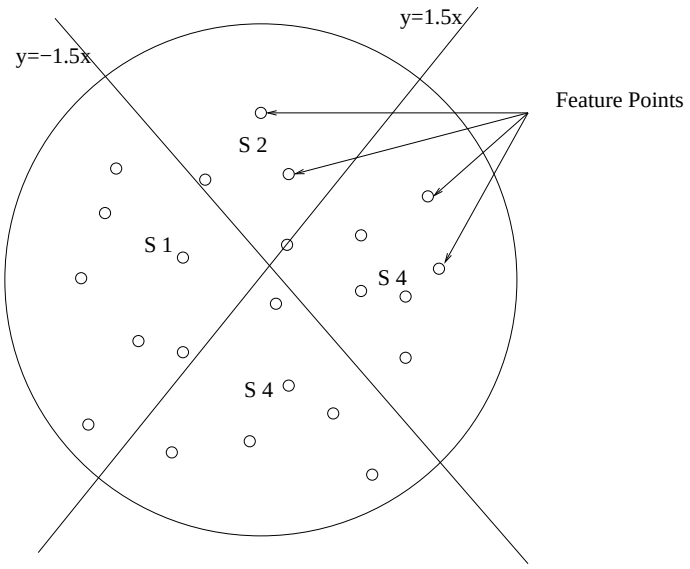

Figure 6.1: Illustration of two correlated feature lines, $y = 1.5x$ and $y = -1.5x$. There feature values of these two lines tends to be the same if the underlaying point sets follows Poisson arrival process.

It then produces the covariance matrix of I, which is denoted as $C_{m\times m}$, where $C_{i,j} = Cov(Column_i(I), Column_j(I))$. An SVD (Singular Value Decomposition) [19] process is applied to decomposite C to three matrix U S and V to remove the correlations between $Column_i(I)$ and $Column_j(I)$ for any pair of i, j. U is the transformation matrix we are most interested in. Its column vectors are eigenvectors, which are orthogonal to each other with unit length. The correlations between each features are reduced to 0 statistically. We will denote U as $(u_1, u_2, ...u_m)$, where each u_i is a column vector. There might be thousands of characteristics in the original dataset, but the column vectors u_i in U are the most significant ones. The weight of each u_i is indicated in the diagonal matrix S and the weight is in decreasing order from $u_1, u_2, ..., u_m$. Further more, the projected value of any vectors to the basis U has the largest variance on u_1, the second largest variance on u_2 and

so on.

6.6 Project Feature Vectors to PCA Basis

In the section 6.5, we obtain an orthogonal basis $U = (u_1, u_2, ..., u_m)$. By inner producting a feature vector v with each u_i, we could get the projected vector $v_{proj(U)}$ of v on basis U, i.e. $v_{proj(U)} = v \cdot (u_1, u_2, ..., u_m)$.

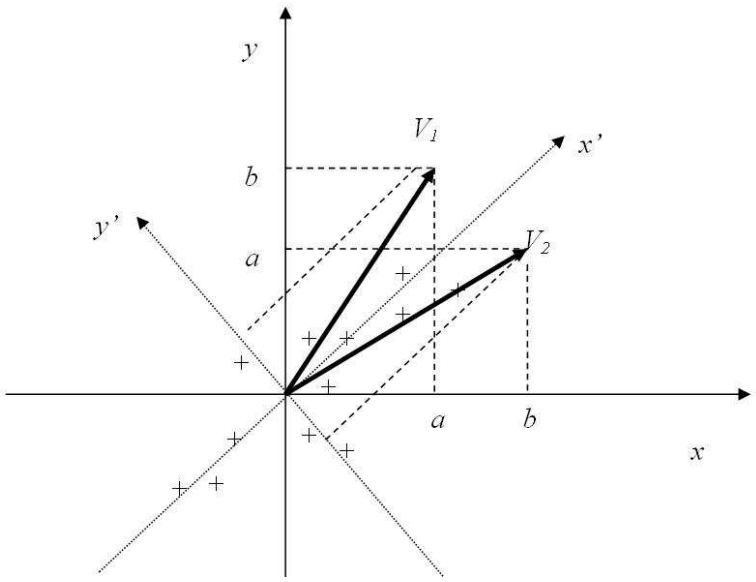

Figure 6.2: Illustration of how PCA amplify the noises. "+" are feature points.

The projected vector $v_{proj(U)}$ maximizes the variance for each projected values. The difference of two input data is well separated by the most different directions using PCA analysis. For example, as Figure 6.2 shows, according to the mechanism of PCA, the distribution of feature points will produce a basis consists of two eigenvectors along x' and y' directions. In fact,

we see along x' direction, there should be no difference between two vectors V_1 and V_2, and during y' direction, there should be $\sqrt{2(a-b)^2} = \sqrt{2}|a-b|$ difference. However, according to the original x-y coordinates system, we find at both direction, namely x direction and y direction, the difference between V_1 and V_2 is both $|a-b|$. The original x-y system didn't catch the features/characteristics of this two vectors. If we employ the rule that the AMAC $f_{verifier}$ will reject a fingerprint/image if the distance to the original copy on any direction is larger than $1.2 * |a-b|$, the PCA version of AMAC verifier will reject the v_2's declaration to be v_1. This shows that the AMAC protocol using PCA is more sensitive to the noises.

Once we obtained $v_{proj(U)}$, we guarantee the robustness of our AMAC protocol such that if the difference between two files is smaller than certain threshold, we produce two projected vectors are within certain distance. In contrast, with very high probability, if two projected vectors are close enough, the original feature vectors are also close.

6.7 Map Projected Vector to Fixed Binary String as AMAC

Without concerning of *fixed length* output bit strings of AMAC protocol requirement, we could have directly output the projected vector as the approximate message authentication code for 2-D point set. Or else, if we just want to have probabilistic comparison functions, we simply measure the distance

of two projected vector and set a threshold.

However, as for the requirement of a AMAC function, we need this step to produce fixed length output. We propose a simple function f_q which quantizes the projected vector to a fixed length string.

Definition 10. *f_q: $\mathcal{M} \rightarrow \{0,1\}^n$ is a quantize function which satisfying following condition: $\forall(x_1, x_2, ..., x_n) \in \mathcal{M}$,if $x_i - \overline{x_i} \geq 0$, output i-th bit of AMAC as 1, otherwise the i-th bit of AMAC is 0. $\overline{x_i}$ is the mean value of x_i.*

Sometimes the number of projected values n in a projected vector might still be large (e.g. $n = 400$) if we chose large number of feature lines, whereas a typical AMAC code would have only 64-bit or 128-bit length. By exploring the properties of PCA, we notice that:

1. The variance of each projected value is strictly decreasing in order. Namely, $Var(x_1) > Var(x_2) > ... > Var(x_n)$.

2. The weight of each eigenvector is strictly decreasing in order. Namely, for a basis $(u_1, u_2, ..., u_n)$, $Weight(u_1) > Weight(u_2) > ... > Weight(u_n)$.

Since the most significant projected values are those leading in the front of the projected vector, the most sensitive bits of the AMAC responding to the noises are also those leading bits. Hence if we have obtained 400-bit binary string from the f_p function and we only want a 128-bit AMAC string, we could chop the first 128 bits from the 400 bits string and output it as the final AMAC string. We employ this idea in our AMAC on 2-D point set

protocol. Another similar idea is to employ only 128 feature lines, so that in the end, there would be just 128 bit AMAC output.

An alternative strategy is to choose only 128 bits from the output 400 bit string. We may keep the selection process as secret.

Chapter 7

Analysis of AMAC for 2-D point sets

In this section, we analyze the robustness and sensitivity properties of the AMAC for 2-D point set protocol.

7.1 Notations

n :	Number of feature lines cutting the template
s :	Number of points in the template
r :	Number of noises
γ :	Range of perturbing noises
k :	Length of AMAC bit string
c_{max} :	$max(\|u_{ij}\|)$, for all i, j in a projection basis.
σ_i^2 :	Variance of a distribution
μ_i :	Mean value of a distribution

7.2 Upper Boundary for Effective Distance Due to Noises

Definition 11. *Given the final output AMAC of length k, the* effective distance *between two projected vectors $v_1 = (x_1, x_2, ..., x_n)$ and $v_2 = (y_1, y_2, ..., y_n)$ is defined as $\|v_1 - v_2\|_e = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_k - y_k)^2}$, where $k \leq n$.*

The purpose of introducing *effective distance* is we only need k bits in the final AMAC, which equals to have only first k projected values in the projected vector. Hence, the effective distance between two projected vectors is limited to the first k projected values.

Definition 12. *We define k-noise as following: the point set X' differs from*

X by only k points, and each of the k points follows one of the noise types defined in Definition 9. A single noise is a k-noise where $k = 1$.

Lemma 1. *If a single noise happens in a template, the effective distance between projected vectors of noisy and non-noisy templates is at most* $2\sqrt{k}n \cdot c_{max}$

Proof. In the worst case, a single noise in a template will affect all the feature lines intersecting with that template.(e.g. a point's missing). Suppose the difference between the two feature vectors is $(\triangle x_1, \triangle x_2, ..., \triangle x_n)$. Since there is only 1 noise, hence every the feature value for every line changes at most by 2, i.e. $|\triangle x_i| \leq 2$ for all i. Assume the basis matrix for PCA transformation is

$$\begin{pmatrix} y_{1,1} & y_{1,2} & ... & y_{1,n} \\ y_{2,1} & y_{2,2} & ... & y_{2,n} \\ ... & ... & ... & ... \\ y_{n,1} & y_{n,2} & ... & y_{n,n} \end{pmatrix}$$

. Hence the effective distance:

$$\begin{aligned} \text{Effective Distance} &= \sqrt{(\triangle x_1 y_{1,1} + \triangle x_2 y_{2,1} + ... + \triangle x_n y_{n,1})^2 + } \\ &\overline{(\triangle x_1 y_{1,2} + \triangle x_2 y_{2,2} + ... + \triangle x_n y_{n,2})^2 + ... + } \\ &\overline{(\triangle x_1 y_{1,k} + \triangle x_2 y_{2,k} + ... + \triangle x_n y_{n,k})^2} \\ &\leq \sqrt{k \cdot (2 \cdot c_{max} \cdot n)^2} \\ &= 2\sqrt{k}n \cdot c_{max} \end{aligned}$$

□

Theorem 1. *If the effective distance between two projected vector is larger*

than δ, there must be at least $\frac{\delta}{2\sqrt{k}n \cdot c_{max}}$ noises between the original two templates.

Proof. Assume the number of noises is r, by Lemma 1, 1 noise will cause at most $2\sqrt{k}n \cdot c_{max}$ effective distance, hence, r noises will cause at most $\delta = r \times (2\sqrt{k}n \cdot c_{max})$ effective distance. Solve r we have $r = \frac{\delta}{2\sqrt{k}n \cdot c_{max}}$. □

7.3 Feature Value Changes Due to A Single Point Noise

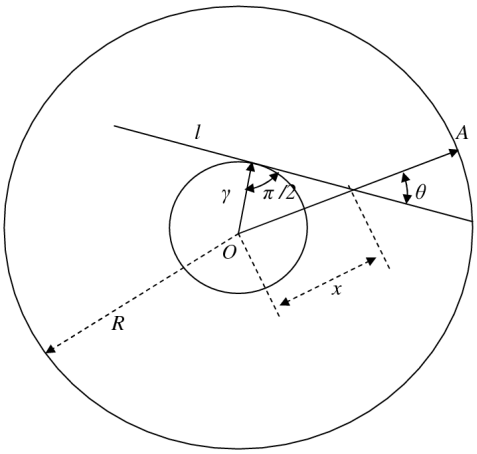

Figure 7.1: Illustration of probability of a random line intersecting with the center circle.

Lemma 2. *The probability p_{out} that a random line doesn't intersect with a circle of radius γ located in the center of a sphere region of radius R is*

$$1 - \frac{2arcsin(\frac{\gamma}{R})}{\pi} - \frac{2\gamma}{\pi R}\sqrt{1 - (\frac{\gamma}{R})^2} \tag{7.1}$$

If we denote γ/R as σ, we could rewrite p_{out} as

$$1 - \frac{2arcsin(\sigma)}{\pi} - \frac{2\sigma}{\pi}\sqrt{1-\sigma^2} \tag{7.2}$$

Proof. The random line is generated by firstly selecting a point inside the sphere region and secondly choosing an angle between 0 to π. Consider a radius $\overrightarrow{OA}$ as Figure 7.1 shows, every line l will have cut any point on the radius with probability $\frac{2\pi x \mathrm{d}x}{\pi R^2}$. Only when the degree of angle between l and $\overrightarrow{OA}$ is smaller than θ and $\gamma < x < R$, will the line cut the center disk. Hence, we have:

$$p_{out} = \int_{\gamma}^{R} \frac{\pi - 2arcsin\frac{\gamma}{x}}{\pi} \cdot \frac{2\pi x \mathrm{d}x}{\pi R^2} = 1 - \frac{2arcsin(\frac{\gamma}{R})}{\pi} - \frac{2\gamma}{\pi R}\sqrt{1-(\frac{\gamma}{R})^2} \tag{7.3}$$

□

This means with probability p_{out}, a noise happened will not affect a specific feature line. In fact, if the noise is of type 2 or 3, which introduce or miss an abstract point, the feature value corresponding to any feature line must be either increasing by 2, or decreasing by 2. Only the first type of noise could achieve the fact that it is possible for a noise which doesn't affect any feature line.

Theorem 2. *The probability that a perturb noise doesn't change any feature line's feature value is* $1 - \frac{2arcsin(\frac{\gamma}{R})}{\pi} - \frac{2\gamma}{\pi R}\sqrt{1-(\frac{\gamma}{R})^2}$.

Proof. We model our perturb noise using the setting in previous lemma and the result preserves. □

Theorem 3. *The probability that a noise affects at least one of the n feature values is at least* $1 - (1 - \frac{2arcsin(\frac{\gamma}{R})}{\pi} - \frac{2\gamma}{\pi R}\sqrt{1 - (\frac{\gamma}{R})^2})^n$.

Proof. We denote $NC()$ as a function, which takes into a line, and outputs the number of feature value is changed associating with this line if a perturb noise happen. Since the feature lines are independent from each other,

$$
\begin{aligned}
&Prob(NC(l_1) = 0 \wedge NC(l_2) = 0 \wedge ... \wedge NC(l_n) = 0) \\
&= Prob(NC(l_1) = 0) \times Prob(NC(l_2) = 0) \times ... \times Prob(NC(l_n) = 0) \\
&= p_{out} \times p_{out} \times ... \times p_{out} \\
&= p_{out}^n
\end{aligned}
$$

Hence,

$$
\begin{aligned}
&Prob(\text{a noise affects at least 1 line}) \\
&\geq Prob(\text{a perturb noise affects at least 1 line}) \\
&= 1 - p_{out}^n
\end{aligned}
$$

. □

For $\sigma = 0.01$, $n = 128$, we have probability of 99.46% that at least 1 feature value will change due to this noise. And the more lines we use, e.g. $n = 400$, the high probability (99.999992%) we guarantee a change in the feature values.

7.4 Probabilistic Lower Boundary for Effective Distance Due to Noises

We denote the vector $(\Delta x_1, \Delta x_2, ..., \Delta x_n)$ as the difference between the original feature vector and the noisy version of feature vector. Since each feature line is independent from each other, we have Δx_i are all independent from each other.

Definition 13. *If there are r noises happened in a template, we define the change of each feature value Δx_i follows a Gaussian distribution, with mean 0, variance $2r\rho$, for some $0 < \rho < 1$.*

Lemma 3. *If $y_1, y_2, ..., y_n$ are n real number such that $y_1^2 + y_2^2 + ... + y_n^2 = 1$, we have $w = \Delta x_1 y_1 + \Delta x_2 y_2 + ... + \Delta x_n y_n$ approximately follows normal distribution with $E(w) = 0$, $Var(w) = 2r\rho$.*

Proof. This is by an approximation that, the sum of a large number of i.i.d random variable follows a normal distribution. Since, Δx_i are all independent from each other, y_i are constant, we have the sum w following a normal distribution. Notice that in PCA transformation basis, each column vector has unit length, i.e. $\sum_{i=1}^{n}(y_{i,k}^2) = 1$ for any column k.

$E(w) = \Sigma E(\Delta x_i y_i) = \Sigma(y_i E(\Delta x_i)) = 0$ and

$Var(w) = \Sigma Var(\Delta x_i y_i) = \Sigma(y_i^2 Var(\Delta x_i)) = 2r\rho\Sigma(y_i^2) = 2r\rho$ by definition 13. □

Lemma 4. *If X is a random variable follows normal distribution with mean*

0, *variance* $2r\rho$, $Prob(X^2 > a^2) = 1 - \texttt{erf}(\frac{a}{2\sqrt{2}r\rho})$, *where* $a > 0$, *function* erf *is the error function.*

Proof. Since X follows normal distribution, $Prob(X > a) = 1 - Prob(X \leq a) = 1 - \frac{1}{2}(1 + \texttt{erf}(\frac{a}{2\sqrt{2}r\rho}))$. $Prob(X^2 > a^2) = 2Prob(X > a) = 1 - \texttt{erf}(\frac{a}{2\sqrt{2}r\rho})$. □

Theorem 4. *With probability* $1 - \texttt{erf}(\frac{a}{2\sqrt{2}r\rho})$, *the effective distance between two projected vectors is at least* a, *if two feature vectors are apart by* r *noises.*

Proof.

$$\begin{aligned}
\text{Effective Distance} = \; & \sqrt{(\Delta x_1 y_{1,1} + \Delta x_2 y_{2,1} + ... + \Delta x_n y_{n,1})^2} \\
& \overline{+(\Delta x_1 y_{1,2} + \Delta x_2 y_{2,2} + ... + \Delta x_n y_{n,2})^2} \\
& \overline{+...} \\
& \overline{+(\Delta x_1 y_{1,k} + \Delta x_2 y_{2,k} + ... + \Delta x_n y_{n,k})^2} \\
\geq \; & \sqrt{(\Delta x_1 y_{1,1} + \Delta x_2 y_{2,1} + ... + \Delta x_n y_{n,1})^2} \\
= \; & \sqrt{X^2}
\end{aligned}$$

Where X follows a normal distribution, with mean 0 and variance $2r\rho$. Hence, we have Effective Distance $\geq \sqrt{X^2} > \sqrt{a^2} = a$, with probability $1 - \texttt{erf}(\frac{a}{2\sqrt{2}r\rho})$ by Lemma 4. □

Corollary 1. *With probability* $(1 - \texttt{erf}(\frac{a}{2\sqrt{2}r\rho}))^k$, *the effective distance between two projected vectors is at least* $\sqrt{k}a$, *if two feature vectors are part with* r *noises.*

Proof. For each $(\Delta x_1 y_{1,i} + \Delta x_2 y_{2,i} + ... + \Delta x_n y_{n,i})^2$ where $i = 1, 2, ..., k$, k is

the length of AMAC, we have probability $1 - \texttt{erf}(\frac{a}{2\sqrt{2}r\rho})$ to ensure it to be larger than some a^2. Hence, the probability for all of the k expressions to be larger than a^2 is $(1 - \texttt{erf}(\frac{a}{2\sqrt{2}r\rho}))^k$.

We have Effective Distance $\geq \sqrt{a^2 + a^2 + ... + a^2} = \sqrt{k}a$ with probability $(1 - \texttt{erf}(\frac{a}{2\sqrt{2}r\rho}))^k$. □

7.5 Preimage Attacks on AMAC

The design requirement of AMAC for 2-D point set is to resist the preimage attacks. Namely, given output y, we are computationally infeasible to find an input x such that $y = AMAC(x)$. It could be viewed as robustness one-way hash functions, where the key, namely the function `AMAC`, is either kept secret or is so complicated that it is infeasible to find a trap door function. Our magic is similar as other hash or MAC functions. However, instead of using shifting and permutation, or mixed/shuffle columns and rows, we use feature line sets as our secret key. We conjecture that, the feature lines combining with PCA technique provide very high computational complexity to resist to preimage attacks. Further more, by training our AMAC function using large amount of homogenous inputs, we are able to amplify the noise level of two similar template inputs, which facilities the detection about sensitivity.

Through extensive experimental comparisons, we find the distribution of AMAC distance follows strictly the normal distribution. Suppose the sample template input space consists of templates which have μ_1 mean AMAC

distance and σ_1^2 distance variance. Assume the AMAC protocol successfully authenticates a noisy version template $\mathcal{T}'$ of $\mathcal{T}$ if $\mathcal{T}'$ is within the range of normal distribution with μ_2 as mean, σ_2^2 as distance variance.

Theorem 5. *The probability of finding a template maps to a AMAC code which is a permissive noisy version of a specific AMAC code is*

$$\frac{e^{-\frac{(\mu_1-\mu_2)^2}{2\sigma_1^2+2\sigma_2^2}}}{\sqrt{8\pi(\sigma_1^2+\sigma_2^2)}}\left\{\texttt{erf}(\frac{\sigma_2^2\mu_1+\sigma_1^2\mu_2}{\sigma_1\sigma_2\sqrt{2(\sigma_1^2+\sigma_2^2)}})-\texttt{erf}(\frac{\sigma_2^2(\mu_1-k)+\sigma_1^2(\mu_2-k)}{\sigma_1\sigma_2\sqrt{2(\sigma_1^2+\sigma_2^2)}})\right\}$$

which is less than

$$\frac{2e^{-\frac{(\mu_1-\mu_2)^2}{2\sigma_1^2+2\sigma_2^2}}}{\sqrt{8\pi(\sigma_1^2+\sigma_2^2)}}$$

Proof. Since we assume the permissive noisy version of AMAC code follows a normal distribution X_2 with μ_2 as mean, σ_2^2 as distance variance and the average AMAC distance between two random templates also follows a normal distribution X_1 with mean μ_1 and variance σ_2^2. With the output AMAC size k, we have,

Prob(a template maps to an AMAC with permissive noise)=

$$\int_0^k \text{Prob}(X_1=x)\cdot\text{Prob}(X_2=x)\text{d}x$$

$$=\int_0^k \frac{e^{\frac{-(x-\mu_1)^2}{2\sigma_1^2}}}{\sigma_1\sqrt{2\pi}}\cdot\frac{e^{\frac{-(x-\mu_2)^2}{2\sigma_2^2}}}{\sigma_2\sqrt{2\pi}}\text{d}x$$

$$=\frac{e^{-\frac{(\mu_1-\mu_2)^2}{2\sigma_1^2+2\sigma_2^2}}}{\sqrt{8\pi(\sigma_1^2+\sigma_2^2)}}\left\{\texttt{erf}(\frac{\sigma_2^2\mu_1+\sigma_1^2\mu_2}{\sigma_1\sigma_2\sqrt{2(\sigma_1^2+\sigma_2^2)}})-\texttt{erf}(\frac{\sigma_2^2(\mu_1-k)+\sigma_1^2(\mu_2-k)}{\sigma_1\sigma_2\sqrt{2(\sigma_1^2+\sigma_2^2)}})\right\}$$

Since $\texttt{erf}(\frac{\sigma_2^2\mu_1+\sigma_1^2\mu_2}{\sigma_1\sigma_2\sqrt{2(\sigma_1^2+\sigma_2^2)}})$ is smaller than 1 and $\texttt{erf}(\frac{\sigma_2^2(\mu_1-k)+\sigma_1^2(\mu_2-k)}{\sigma_1\sigma_2\sqrt{2(\sigma_1^2+\sigma_2^2)}})$ is larger than -1, hence we have the original expression $< \frac{2e^{-\frac{(\mu_1-\mu_2)^2}{2\sigma_1^2+2\sigma_2^2}}}{\sqrt{8\pi(\sigma_1^2+\sigma_2^2)}}$ □

Chapter 8

Experiments on AMAC for 2-D point sets

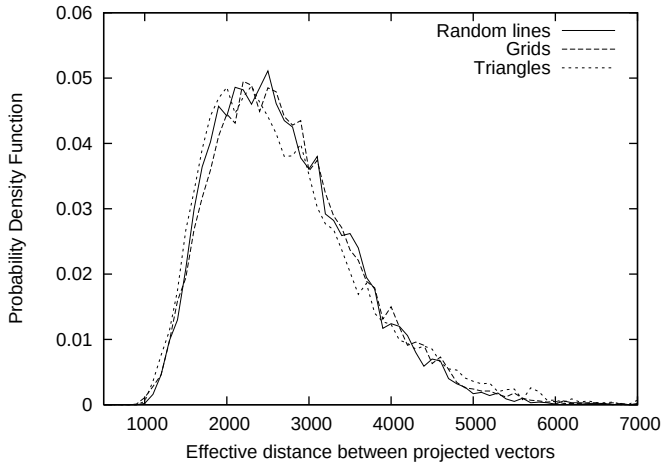

Figure 8.1: Probability density function for effective distance between two random templates. Different feature line sets are used: random lines, grid lines and triangle lines.

8.1 Experiment Settings

In following experiments, we set the template space to be a square of size 100 by 100. 10000 feature points are distributed on the space using binary approximation of Poisson process. 405 random feature lines are used and they are independent with each other. We assume the final output AMAC is of 128 binary bit length. The noise types are as defined in Definition 9. The probability of noises is 50% for the 1st kind of perturbing noise and 50% for the 2nd and 3rd type of noises. We assume that if the perturbing noise happens, a point will move to a random point within radius γ only. We assume a newly introduced point may appear at any location, despite in reality, for example, in fingerprint template, two minutiae points cannot be too close to each other. All the probability density experiments are based on 10000 sample output.

Projection Basis The projection basis consists of two types, 1. we use 405 random feature lines and 10000 point sets, the points are generated using approximation Poisson process; 2. we use 405 random feature lines and 10000 homogeneous point sets, the points sets are generated as the noisy version of one point set, with 30% of points either perturbing with 1% of the length of the space. Hence the first set of projection basis is more general and suitable for the point sets which follows global patterns; the second set of projection basis is more specific to a set of homogenous data which follows specific patterns.

8.2 Choosing Feature Lines

We use different sets of feature lines to check whether choosing different types of feature lines will affect the result very much. We choose three sets of lines: 1. *random* which consists of 405 random lines, 2. *grids* which consists of 202 evenly distributed horizontal lines and 203 evenly distributed vertical lines, 3. *triangle* which consists of 135 evenly distributed +60 degree lines, 135 evenly distributed -60 degree lines and 135 evenly distributed horizonal lines. Using the first type of basis, it turns out that the effective distance distributions are about the same regardless the choice of lines. Hence, the lines could be regular, could be random, the result is not much affected, See Figure 8.1. However, it doesn't mean individual choice of lines is not important. Recall that the feature lines will determine the feature values and PCA process generate projection basis according to these values.

8.3 Choosing Different Noisy Levels

Using the first type of basis, we measure the relationship between noisy level and the mean/variance of difference in AMAC code, as shown in Figure 8.2 and Figure 8.3. When the noisy points consists of 1% of the whole point set points, on average, there would be 4.86 bits difference of a 128-bit AMAC code with small variance 3.30. If there are 50 percent of noises, the mean AMAC distance is scaled up to 39.83 bits and the variance is also brought up to 23.78.

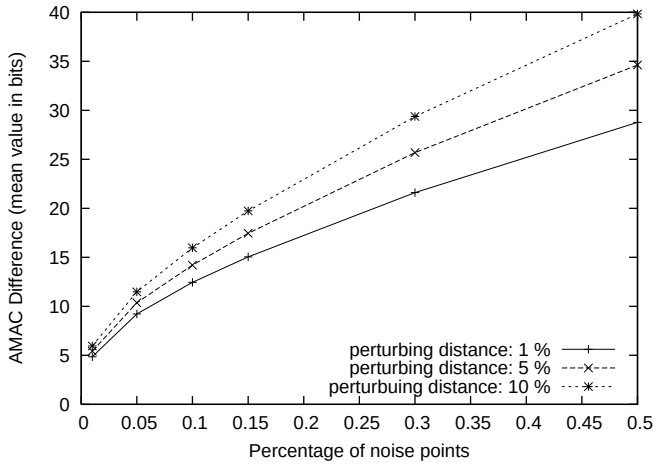

Figure 8.2: Mean AMAC bit difference with difference noise levels and perturb ranges

8.4 Average AMAC Distance between Templates

For any two random copies of point set templates, we have the AMAC distance distribution using first type of basis as shown in Figure 8.4. The average bit difference (the right curve) between two AMACs is 64 bits, which is fairly reasonable for an AMAC code with length 128 bit. Using the same projection basis, we produce the AMAC distance distribution for two noisy version AMACs. We find if 30% points are affected by noises, including 15% percent of noise type 1 and 15% of noise type 2 and 3, we will have 21 bit AMAC difference in mean(the left curve), assuming the perturbing distance is within 1% range.

The distribution of the AMAC distance is helpful in calculating preimage attack complexity. Based on analysis in section 7.5, using a normal distri-

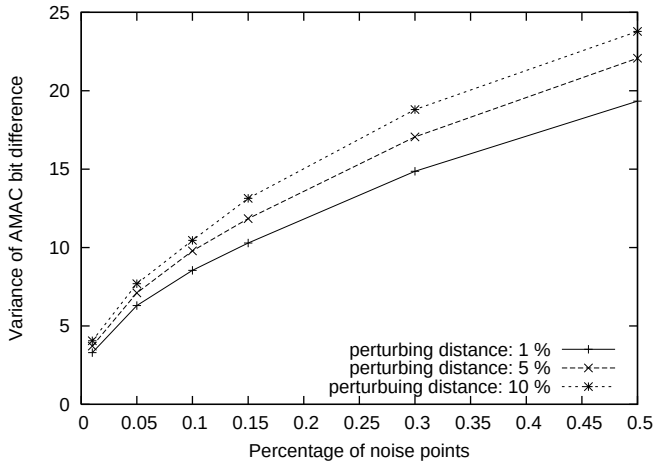

Figure 8.3: Variance of AMAC bit difference with difference noise levels and perturb ranges

bution approximation, we find given an AMAC, the probability to find a template mapping to similar MAC(define similar to be the noisy settings mentioned above) is less than 1.49×10^{-11}. If we set the permitted noise bound tighter, which only allows no more than 1% noises, the probability of finding a template mapping to a MAC reduces to 1.10×10^{-24}.

It would also be interesting to see what happens if we train our AMAC protocol using these homogeneous templates. As shown in Figure 8.5 (the right side curve), if we generate the projection basis according to these homogeneous templates, which are noisy versions of some template X. The average AMAC distance between any two copies shifts from the left side curve to the right side curve, and the mean distance would become 64 again. This means, if we train the AMAC protocol with special pattern data, the AMAC protocol is able to amplify the difference level as if they are two random copies of point set templates. This is very useful technique in distinguishing

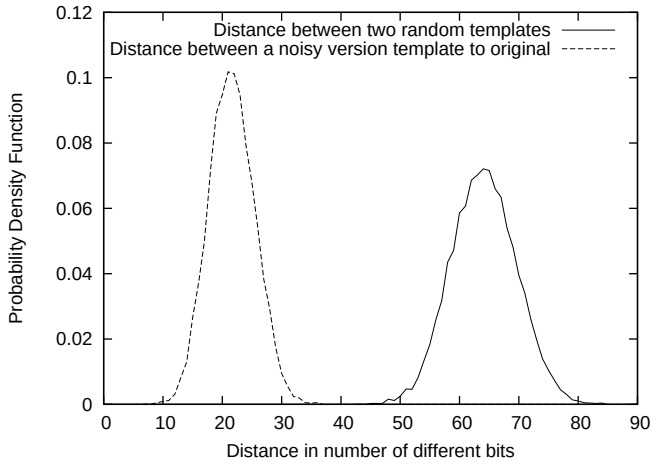

Figure 8.4: Probability density function for AMAC distance between (1) any random two templates, (2) a noisy version to an original template. 30% of the points are affected by noises, perturbing distance is within 1% of the range size.

homogeneous data sets, such as distinguishing handwritten images.

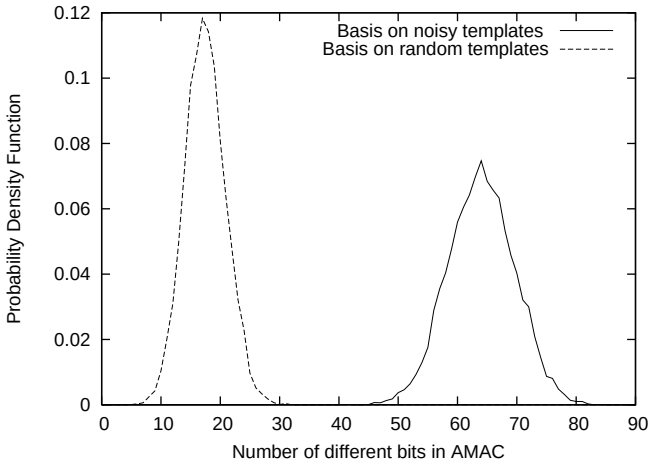

Figure 8.5: AMAC distance distribution with different projection basis: (1) using basis generated among noisy templates (2) using basis generated among random templates

Chapter 9

Conclusion and Future Works

9.1 Conclusion

A fingerprint is typically represented as a set of minutiae which consists of 2-D points in authentication. In this thesis, we explore ways of extracting consistent bits from such point set templates for robust authentication. This thesis consists of two parts. In the first part, we show analysis of an existing method: *Fingerprint vault* [6] scheme. It produces a public sketch for authentication by adding chaff points into the original minutiae set. By observing that a chaff point which comes later tends to have smaller free area, we propose a method to find out the original minutiae from the vault on average 2192.7 times faster than a brute-force attacker.

In the second part, we inspired by a robust authentication scheme-AMAC for robust image authentication, where the message authentication informa-

tion is kept as a small fixed length string. We propose an *AMAC for 2-D point set* which extracts positioning information between points using feature lines and projects this information to appropriate Euclidean space using PCA and a quantizing function. Given such a 128-bit length AMAC code, to successfully launch a pre-image attack requires more than 2^{81} trials. We could further construct secure sketch using this extracted information since it preserves locality information and yet sensitive to large tampers.

9.2 Future Works

In our implementation of approximate message authentication code for 2-D point set templates, we use a quantize function which maps the projected vectors to a binary code. It is a very simple mapping function, which outputs 1 if the projected value is above average value and outputs 0 otherwise. We believe there are better quantize functions which keep more information and produce more sensitive AMAC codes. Further more, since our projected vectors shows locality preserving property, we may send them to secure sketches to extract consistent bits. It is interesting to explore them in the future work. Other than this, whether we could use other methods to generate feature values instead of feature lines is also open for discussion.

We also feel a need to extract real point sets from fingerprint library or handwritten images to study the effects of both attack and the *AMAC for 2-D point set* scheme. With the specific patterns of the point set distributions, we expect to find more reality and practical results of our works.

Bibliography

[1] Fvc2004 databases. *http://biometrics.cse.msu.edu/fvc04db/index.html*.

[2] Arce, G., Xie, L., and Graveman, R. Approximate image authentication codes. In *Proc. 4th Annual Fedlab Symp. on Advanced Telecommunications/Information Distribution* (2000).

[3] Bishop, M. *Computer Security: Art and Science*. Addison-Wesley, 2003.

[4] Chang, E.-C., and Li, Q. Small secure sketch for point-set difference. *Cryptology ePrint Archive, Report 2005/145* (2005).

[5] Chang, E.-C., Shen, R., and Teo, W. Finding the original point set hideen among chaff. In *Proc. ACM Sym on Information, Computer and Communications Security* (2006).

[6] Clancy, T. C., Kiyavash, N., and Lin, D. J. Secure smartcard-based fingerprint authentication. In *ACM SIGMM workshop on Biometrics methods and applications* (2003), pp. 45–52.

[7] DiCrescenzo, G., Graveman, R., Arce, G., and Ge, R. A formal security analysis of approximate message authentication codes. In *Proc. CTA Comm. and Networks* (2003).

[8] Dodis, Y., Reyzin, L., and Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Eurocrypt'04* (2004), pp. 523–540.

[9] Graveman, R., and Fu, K. Approximate message authentication codes. In *Proc. 3rd Annual Fedlab Symp. on Advanced Telecommunications/Information Distribution* (1999).

[10] Jolliffe, I. *Principal Component Analysis*. Springer-Verlag, 1986.

[11] Juels, A., and Sudan, M. A fuzzy vault scheme. In *IEEE Intl. Symp. on Information Theory* (2002).

[12] Juels, A., and Wattenberg, M. A fuzzy commitment scheme. In *ACM Conf. on Computer and Communications Security* (1999), pp. 28–36.

[13] Lin, S., and Costello, D. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, 2004.

[14] Linnartz, J.-P. M. G., and Tuyls, P. New shielding functions to enhance privacy and prevent misuse of biometric templates. In *AVBPA 2003* (2003), pp. 393–402.

[15] Minsky, Y., Trachtenberg, A., and Zippel, R. Set reconciliation with nearly optimal communications complexity. In *ISIT* (2001).

[16] Palasti, I. On some random space filling problems. *Publ. Math. Inst. Hung. Acad. Sci. 5* (1960), 353–359.

[17] Renyi, A. On a one-dimensional problem concerning random space-filling. *Publ. Math. Inst. Hung. Acad. Sci. 3* (1958), 109–127.

[18] Stalling, W. *Cryptography and Network Security: principle and practices*. Prentice Hall, 2003.

[19] Strang, G. *Introduction to Linear Algebra, 3rd Edition*. Wellesley-Cambridge Press, 1998.

[20] van LINT., J. *Introduction to Coding Theory*. Springer-Verlag, 1982.

[21] Xie, L., Arce, G., and Graveman, R. Approximate image message authentication codes. In *IEEE Trans. on Multimedia* (2001), pp. 242–252.

[22] Zhang, W., Chang, Y.-J., and Chen, T. Optimal thresholding for key generation based on biometrics. *Int. Conf. on Image Processing* (2004).